Sommaire

Description de l'algorithme et utilisation

Description

L'algorithme a deux utilisations différentes :

- La première est lorsqu'un seul fichier est passée en argument du programme, affiche pour chaque ligne de texte non vide du fichier, apparaissant au moins deux fois dans le fichier, les numéros des lignes où elle apparaît et le contenu de la ligne.
- La seconde est lorsque au minimum deux fichiers sont passés en argument du programme, affiche pour chaque ligne de texte non vide apparaissant au moins une fois dans tous les fichiers, le nombre d'occurrences de la ligne dans chacun des fichiers et le contenu de la ligne.

Utilisation

- Appel à exécutable "./lnid" puis les fichiers et/ou les options peuvent être mise à n'importe quel endroit.
  - Liste des options

-h	–help	Affiche sur la sortie standard la description du programme et l'utilisation de celui-ci
-u	–uppercase	Transforme tous les caractères traités et étant des caractères minuscules en majuscules
-f [option]	–filter=[option]	Applique l'un des douze tests d'appartenance à une catégorie de caractères is… de l'en-tête standard <ctype.h> passée en argument à filter (Remplacer [option] par le nom du filtre comme "isalnum")

- Si l'option help est appelée, elle s'exécutera puis le programme s'arrêtera proprement.
- L'option uppercase ne s'applique seulement sur les caractères alphabétiques minuscules.
- Dans le cas où plusieurs filtres sont appelés, seul le dernier filtre appelé sera appliquer.

Tous les arguments passés à l'exécutable et qui ne sont pas des options prises en charge doivent être un fichier valide sinon une erreur à l'exécution se produira.

Implémentation

Problématiques

Le programme avait plusieurs obligations et problématique pour son fonctionnement.

1. Le programme devait pouvoir lire sur un fichier une chaîne de caractère dont on ne connaissait pas sa longueur.
2. Il devait aussi pouvoir enregistrer des occurrences de chaque chaîne de caractère selon le nombre de fichiers que l'on ne connaît qu'à l'exécution ou enregistrer des numéros de lignes qui lier à un fichier qui peut avoir un nombre de lignes très petit ou très long.
3. Il devait donc pouvoir enregistrer toutes ses données en prenant le moins de temps lors de recherche.
4. Et faire cela en étant le plus rapide et en prenant le moins d'espace mémoire

Solutions

Mon implémentation se découpe en 5 modules, plus la main.

Les modules

1. Les deux premiers modules sont le module hashtable et holdall qui sont ceux fournis et étudier en cours et qui n'ont pas été modifier.
2. Le troisième module est le module opt qui a pour but de gérer les options, il possède trois fonctions :
    1. opt_gen qui alloue les ressources pour gérer une option.
    2. opt_dispose qui libère les ressources allouées pour une option.
    3. opt_init qui elle initialise les options et sépare les arguments entre les options et les fichiers.
3. Le quatrième est un module polymorphe de tableaux dynamique da qui contient des références vers des objets.
    1. Il possède les fonctions classiques telles que da_empty, da_add, da_dispose, da_ref qui renvoie la référence d'indice i du da et da_lenght.
    2. La dernière fonction est da_dispose_element qui libère les zones mémoires pointées par les références contenues dans da.
4. Le dernier module est une ds qui est un module de chaîne de caractère dynamique qui ne stocke aucune donnée (aucune allocation n'est à effectuer sur l'élément à ajouter).
    1. Il possède les mêmes fonctions classiques que le module da, donc ds_empty, ds_dispose, ds_add, ds_ref, ds_length.
5. Lorsque l'ont créé un da ou un ds une zone mémoire d'une certaine taille est allouée la taille de cette zone mémoire pour da ou ds sont une macroconstante nommé DA___CAPACITY_MIN et DS___CAPACITY_MIN où

DA_CAPACITY_MIN vaut 4, car après plusieurs tests dans un même texte c'est la récurrence moyenne d'une ligne, il y aura de la réallocation obligatoire sur des textes qui assez gros comme les misérables où dans ce texte des lignes apparaissent plus de 20 fois, mais j'ai choisi 4 comme valeur initiale puisque cela permet de sauver de la mémoire tout en restant efficace sur des petits textes. DS___CAPACITY_MIN lui est à 32, car 32 caractères par ligne permet d'être assez efficace sur des textes avec de longue ligne sans perdre en efficacité sur des textes avec des petites lignes.

Le main Dans le main se trouve une nouvelle structure nommée cntxt qui sera le contexte d'utilisation pour nos fonctions des différents modules ou des fonctions du main. Elle contient 3 champs, les deux premiers sont liés aux options qui sont déclarées ensuite.

1. Le champ filter prend la fonction de filtre passé en argument (ex : l'option –filter=isalnum alors le champ filter du contexte prendra alors la fonction isalnum de la bibliothèque <ctype.h>).
2. Le champ transform lui prend la fonction liée à l'option –uppercase, il prend donc la fonction toupper si l'option est passée en argument à l'exécution.

Le dernier champ lui est un da qui suite à l'appel de opt_init contiendra les arguments passés au programme qui ne sont pas des options. Ainsi, si une option est mal écrite, alors une erreur lors de l'exécution se produira.

Ensuite sont déclarées les options par des appels à opt_gen, qui prend comme paramètre les deux noms de l'option, sa description ainsi que sa fonction qui permet d'initialiser le champ du contexte lié.

Puis toutes nos structures qui seront utiliser sont allouées comme hashtable, holdall, un da et un ds ainsi qu'un autre da pour la structure cntxt.

Le corps du main est une boucle qui appelle la fonction addline qui enregistre une ligne du fichier ouvert en lecture est l'enregistre dans un ds. Puis une recherche dans la table de hachage se fait et selon le résultat soit la recherche est positive donc le compteur lié à la ligne est incrémentée, soit si nous somme sur le premier fichier la ligne est ajoutée à la table et sont le tableau de compteur est ajouté lui aussi.

Ensuite, pour chaque chaîne contenue dans la table, si elle respecte la condition selon s'il y a un ou plusieurs fichiers, alors elle est affichée précéder des éléments du tableau de compteur qui lui sont liés.

Pour finir, une libération de toute la mémoire allouée est effectuée.

Exemple d'exécution

Exécution sur le texte bobdylan_iwysb :

- Résultat

```
florent@florent-VirtualBox:~/Documents/GitHub/Project-Al/nbline$ ./lnid ./test/bobdylan_iwysb.txt
10,20,34,44     Honey, I want you
9,19,33,43      I want you so bad
8,18,32,42      I want you, I want you
```

- Avec Valgrind

```
==7089== Memcheck, a memory error detector
==7089== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7089== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==7089== Command: ./lnid ./test/bobdylan_iwysb.txt
==7089==
10,20,34,44     Honey, I want you
9,19,33,43      I want you so bad
8,18,32,42      I want you, I want you
==7089==
==7089== HEAP SUMMARY:
==7089==     in use at exit: 0 bytes in 0 blocks
==7089==   total heap usage: 395 allocs, 395 frees, 36,954 bytes allocated
==7089==
==7089== All heap blocks were freed -- no leaks are possible
==7089==
==7089== For lists of detected and suppressed errors, rerun with: -s
==7089== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

- Et time

```
10,20,34,44     Honey, I want you
9,19,33,43      I want you so bad
8,18,32,42      I want you, I want you

real    0m0,001s
user    0m0,001s
sys     0m0,000s
```

Les misérables

- Ceci n'est qu'un échantillon du résultat donné dû à la longueur du résultat

```
48259,50203     Jean Valjean continua:
47312,49662,49666       --Je le sais.
47295,49613     --Apres?
47272,47300     --Ecoute.
45190,46607     Filles-du-Calvaire.
44957,47004     Jean Valjean garda le silence.
44917,44986     Thenardier reprit:
42956,48972     formidable.
42935,45896     fin.
42271,42321     Le pere repondit:
42259,44985     eux.
42065,47751     combat.
42007,42015,42020       C'est la faute a Rousseau.
42005,42013,42018,42047 C'est la faute a Voltaire,
41046,47266     --Je le connais.
40205,40210,40215,40220,40225,40230,40235,40240,40359,40364,40369,40374          _Lon la._
40204,40209,40214,40219,40224,40229,40234,40239,40358,40363,40368,40373          _Ou vont les belles
 filles,_
40087,40311,41958       Gavroche repondit:
39679,44545     M. Gillenormand, rue des Filles-du-Calvaire, no 6, au Marais."
39612,39841     Nous serons ce soir rue de l'Homme-Arme, no 7. Dans huit jours nous
39611,39840     "Mon bien-aime, helas! mon pere veut que nous partions tout de suite.
39343,39358     --Allez-vous-en, ou je fais sauter la barricade!
39284,48804     le pave.
39217,41696,41708,42625,45345   barricade.
39150,40832     Enjolras reprit:
39133,39215     --Feu! dit la voix.
39098,39336,41994       fumee.
38856,40400,40808       peuple.
38732,41429     l'armee.
38473,42633     --Javert.
38463,45112     --Qui etes-vous?
38047,41518     rue.
38000,41819     Chanvrerie.
37988,45244     suivait.
37705,38082     Grantaire repondit:
37687,43322     Corinthe.
37466,49702     --Qui ca?
37423,37429,37435,37441          _Je n'ai qu'un Dieu, qu'un roi, qu'un liard et qu'une botte._
```

- Valgrind

```
==9679== Memcheck, a memory error detector
==9679== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==9679== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==9679== Command: ./lnid ./test/lesmiserables.txt
==9679==
==9679==
==9679== HEAP SUMMARY:
==9679==     in use at exit: 0 bytes in 0 blocks
==9679==   total heap usage: 557,837 allocs, 557,837 frees, 87,528,639 bytes allocated
==9679==
==9679== All heap blocks were freed -- no leaks are possible
==9679==
==9679== For lists of detected and suppressed errors, rerun with: -s
==9679== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

- Time

```
real    0m0,115s
user    0m0,106s
sys     0m0,009s
```

Fichier de 120 mo

- Valgrind

```
==28949== Memcheck, a memory error detector
==28949== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==28949== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==28949== Command: ./lnid ./test/120mo.txt
==28949==
==28949==
==28949== HEAP SUMMARY:
==28949==     in use at exit: 0 bytes in 0 blocks
==28949==   total heap usage: 1,500,025 allocs, 1,500,025 frees, 3,387,390,309 bytes allocated
==28949==
==28949== All heap blocks were freed -- no leaks are possible
==28949==
==28949== For lists of detected and suppressed errors, rerun with: -s
==28949== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

- Time

```
real    0m2,201s
user    0m2,048s
sys     0m0,148s
```

Figure 1: Capture d'écran du 2023-05-07 16-34-15.png

Comparaison avec le programme du groupe Haddag Edouard, Théo Renaux Verdiere

- Valgrind

```
==30496== Memcheck, a memory error detector
==30496== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==30496== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==30496== Command: ./lnidedth ./test/120mo.txt
==30496==
./test/120mo.txt
==30496==
==30496== HEAP SUMMARY:
==30496==     in use at exit: 0 bytes in 0 blocks
==30496==   total heap usage: 600,039 allocs, 600,039 frees, 131,795,284 bytes allocated
==30496==
==30496== All heap blocks were freed -- no leaks are possible
==30496==
==30496== For lists of detected and suppressed errors, rerun with: -s
==30496== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

- Time

```
real    0m2,393s
user    0m2,186s
sys     0m0,205s
```

Problème durant la conception

1. Le module ds est dû au fait que je n'ai pas réussi à utiliser efficacement le module da pour la lecture de ligne, cela me posait des problèmes de mémoires qui sont pareillement réglées.
2. Le module opt est lui le module qui fut le plus compliqué à faire du au fait que celui-ci est totalement utilisable en dehors de ce projet et que beaucoup de cas sont à traiter.
3. Mon plus gros problème est une sur-utilisation de la mémoire que je n'arrive pas ni à expliquer ni à corriger. Ce problème de mémoire est un facteur de 28,30 sur la taille du texte.

Attention, avec des fichiers trop gros ou en trop grand nombre, dû à la surutilisation de mémoire, cela peut résulter par un crash de l'ordinateur.

Conclusion

- Mon programme est donc formé de 3 modules (hors main) dont 2 sont totalement polymorphes et réutilisables dans de futur projet.
- Le temps d'exécution de mon programme est suffisant en comparaison avec celui de Haddag Edouard et Théo Renaux Verdiere qui lui a beaucoup plus d'option, mais comme je m'en dotais mon programme à un réel problème en termes de gestion de mémoire.
- Mon programme a pour le moment 2 grands axe d'améliorations :
  - Le problème d'utilisation de la mémoire qui vient très probablement de ma manière de lire et de stocké temporairement les caractères d'une ligne.
  - Ma gestion d'option : possibilité d'ajouter une fonctionnalité pour essayer de reconnaître si l'argument passer en paramètre est une option simplement mal tapée pour lors de l'erreur aidée l'utilisateur.
- Je tiens à remercier Haddag Edouard, Théo Renaux Verdiere, Louis Dumontier Louis et Salles Théo pour leur aide à la résolution de problème sur mon programme ainsi que de m'avoir fourni leurs exécutables pour faire des comparaisons.